CUSTOMER CHURN ANALYSIS

Submitted by:

Puram Nagaraju
Batch-1836

PROBLEM STATEMENT

Customer churn is when a company's customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals. Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can prioritize focused marketing efforts on that subset of their customer base. Preventing customer churn is critically important to the telecommunications sector, as the barriers to entry for switching services are so low.

You will examine customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models.

INTRODUCTION TO CUSTOMER CHURN

Definition:

Customer churn or customer attrition is the phenomenon where customers of a business no longer purchase or interact with the business. A high churn means that a higher number of customers no longer want to purchase goods and services from the business. Customer churn rate or customer attrition rate is the mathematical calculation of the percentage of customers who are not likely to make another purchase from a business.

Customer churn happens when customers decide to not continue purchasing products/services from an organization and end their association. Customer churn can prove to be a roadblock for an exponentially growing organization and a retention strategy should be decided in order to avoid an increase in customer churn rates.

Importance of predicting Customer Churn:

The ability to be able to predict that a certain customer is at a very high risk of churning, while there is still some time to do something significant about it, itself represents a great additional potential revenue source for any business.

- It's a fact acquiring new customers is a costly affair but losing the existing customers will cost even more for the business or the organization. As existing paying customers are usually returning customers who if happy will purchase repeatedly from your brand.
- The competition in any market is on a rise and this encourages organizations to focus not only on new business but also on retaining existing customers.
- The most essential step towards predicting customer churn is to start awarding existing customers for constant purchases and support.
- An entire customer journey leads to customer churn and not just a few incidents. Due to the priority of avoiding customer churn, organization's should start offering incentives on purchases of these soon-to-churn customers.

Libraries used:

- import NumPy as np
- import pandas as pd
- import matplotlib. pyplot as plt
- import seaborn as sns

Collection of Data:

```
Customer_churn_data = pd.read_csv("customer_churn.csv")
```

```
Customer_churn_data.head()
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows × 21 columns

Here we have collected the data and predicted customer churn rate which is converted into .csv format and this .csv file is converted into dataframe to read the file. In the above collected data we can see that we have 21 columns and also the data is the combination of the numerical and categorical values that means further "Data Cleaning" has to be done for the good accuracy of the model.

Size of the data:

We need to enter the following code to get the size of the data in the notebook

Customer_Churn_data.shape

As we can see, we have 7043 rows and 21 columns. Further we have to check the null values data types and also statistical analysis of the data and then we have to analyze to drop any unnecessary columns present in the dataset.

```
Customer_churn_data.isnull().sum()

customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

```
Customer_churn_data.dtypes

customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

Now here from the above information we can analyze that our dataset doesn't have any null values and also our dataset contains the data which is the combination of "object", "float" and "int" data types but also the column "Total Charges" is seems to be with "int" or "float" data types but it is "Object data type" so we have to convert it into "float" data type or "int" datatype.

Now we will check the value_counts of the column and proceed with processing of the column proceeding with some changes.

```
Customer_churn_data["TotalCharges"].value_counts()
```

```
             11
20.2         11
19.75         9
20.05         8
19.9          8
             ..
6849.4        1
692.35        1
130.15        1
3211.9        1
6844.5        1
Name: TotalCharges, Length: 6531, dtype: int64
```

I can say that there is the data in the columns which is "" and so because of this we are not able to convert into float and so we will replace it with the "nan" data and then we will drop the null values again.

```
Customer_churn_data['TotalCharges']  = Customer_churn_data['TotalCharges'].replace(' ',np.nan)
```

```
Customer_churn_data.isnull().sum()
```

```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

Here we have converted or replaced the gaps in the column with “nan” values and then after we have checked the null values which we can see that in the column “Total Charges” we can see that there are 11 nan values which have to be filled further and convert to float datatype to proceed for further preprocessing.

```
Customer_churn_data['TotalCharges'] = Customer_churn_data['TotalCharges'].astype('float')
```

```
Customer_churn_data.dtypes
```

```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges        float64
Churn                object
dtype: object
```

Here we have successfully converted the column "Total Charges" from object datatype to float data type and so now we can proceed filling with the mean values and then check for the null values again.

```
Customer_churn_data['TotalCharges'] = Customer_churn_data['TotalCharges'].fillna(Customer_churn_data['TotalCharges'].mean())
```

Checking the null-values again :

```
Customer_churn_data.isnull().sum()
```

```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

Here we have filled the null values with “. fillna” method while filling the spaces with “mean” values.

Dropping the unnecessary columns:

```
Customer_churn_data = Customer_churn_data.drop(columns = 'customerID')
```

The column “Customer_id” has no relationship with our label and thus we have dropped the column and proceed further for our model building.

Statistical analysis of the dataset:

We can observe that in most of the above numerical columns the mean is little greater than standard deviation except for the column “Senior Citizen”.

By the observations and the statistical analyzed data, it seems that the dataset seems to be perfect and also there are no negative/invalid values present.
Also, we can observe that “mean” value is greater than “median”ie50% quantile in the columns “Tenure” and “Total Charges” which indicates that these columns are skewed towards right ieThese columns have positive skewness.

Also, we can observe that the column “Monthly Charges” has the “mean value” smaller than “median value “ie., 50% quantile, which indicates that the column is skewed towards left which means this column has negative skewness.

Also, we can observe that there is large difference between 75% quantile and mx quantile which indicates that the data has outliers within it which have to be treated in further processing, missing it may affect our model accuracy.

Checking the count for our label column “Churn”:

Here we can see that we have taken the “value_counts” for the column “Churn” and also we have converted it into dataframe and also which clearly indicates that the number of customers who have churned is indicated by the category “Yes” and the number of customers who have not churned is indicated by the column “No”.

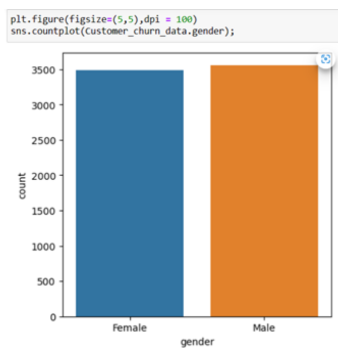
Now we will plot heatmap for null-values:
Here we can see that no null-values can be seen in heatmap.

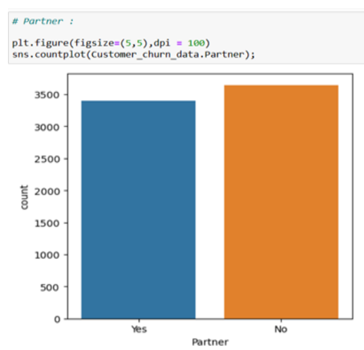
Now let’s visualize our columns individually (“Univariate analysis”) and also visualizations of the columns with our label (“Bivariate analysis”) and even “correlation”.

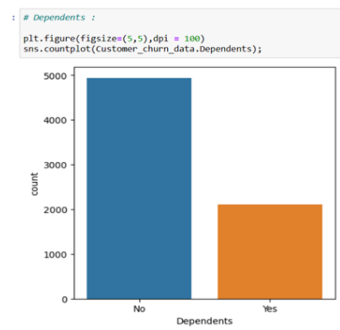
VISUALIZATION:

Univariate analysis:

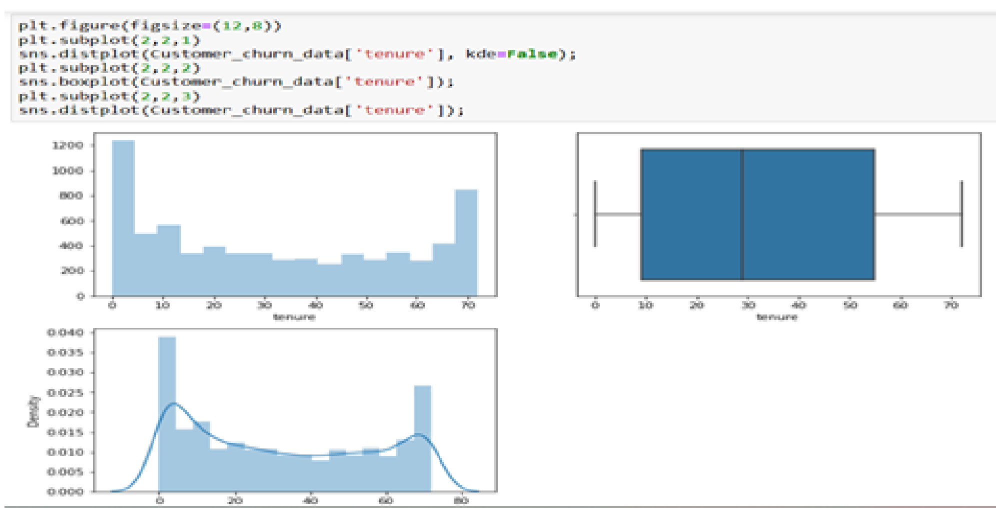
Gender **Partner** **Dependents**





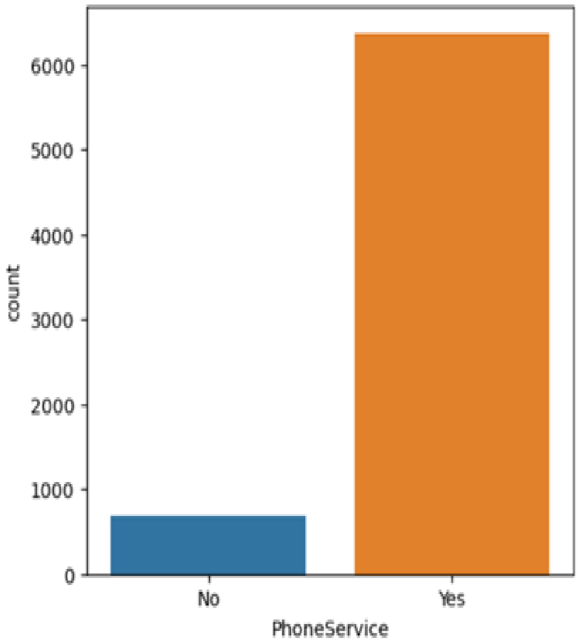


Tenure

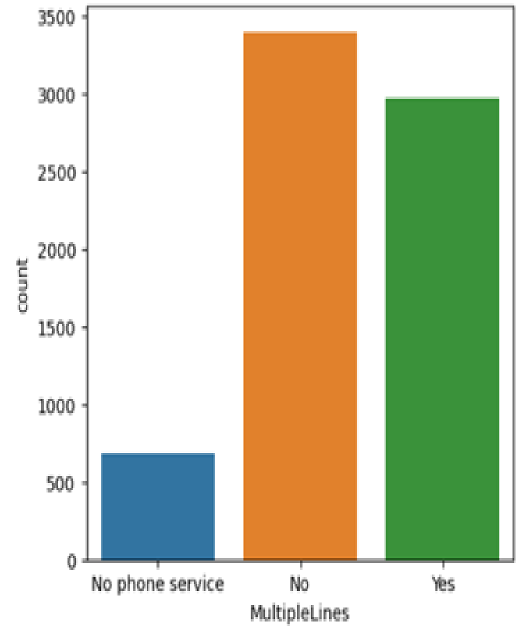


Phone service **Multiple lines**

```
# PhoneService :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.PhoneService);
```

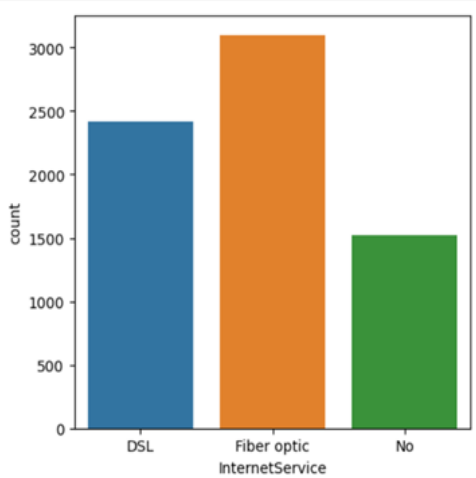



```
# MultipleLines :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.MultipleLines);
```



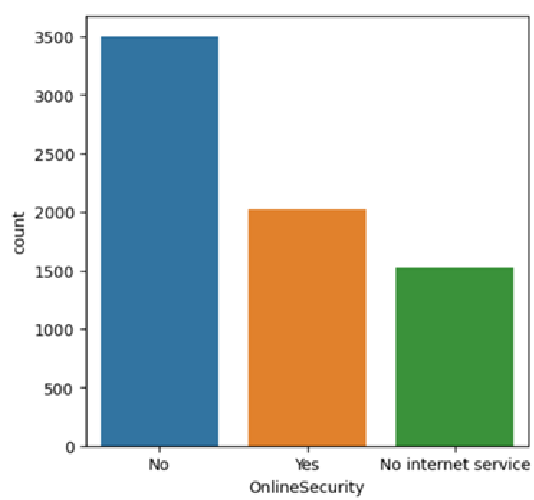
Internet Service

```
# InternetService :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.InternetService);
```



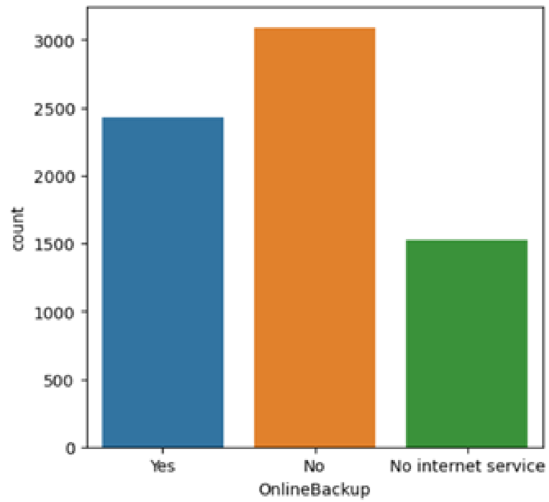
Online Security

```
# OnlineSecurity :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.OnlineSecurity);
```



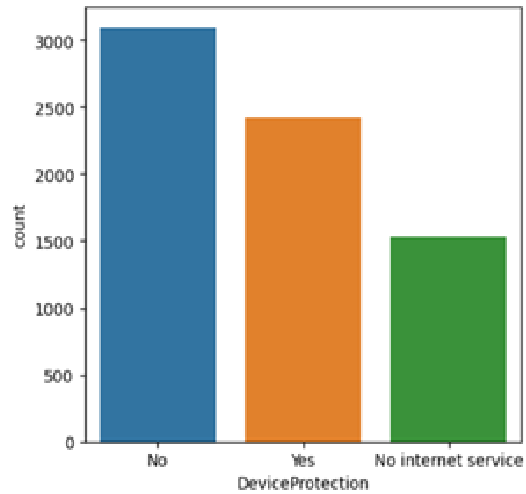
Online Backup

```
# OnlineBackup :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.OnlineBackup);
```



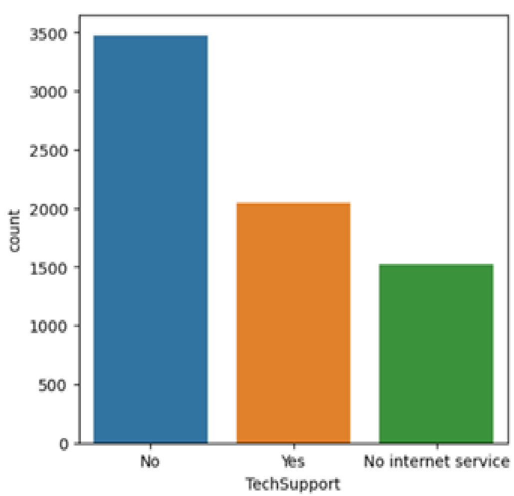
Device Protection

```
# DeviceProtection :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.DeviceProtection);
```



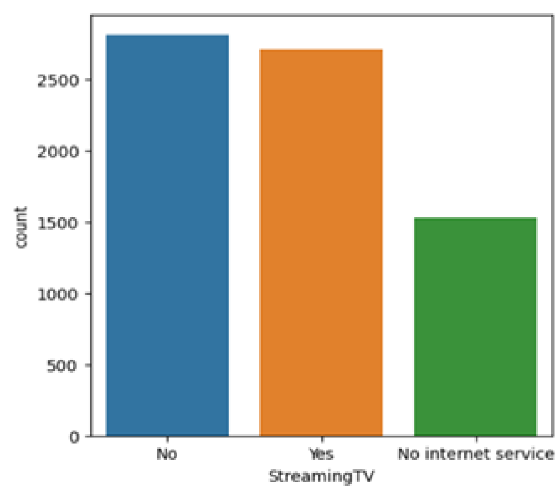
Tech Support

```
# TechSupport :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.TechSupport);
```



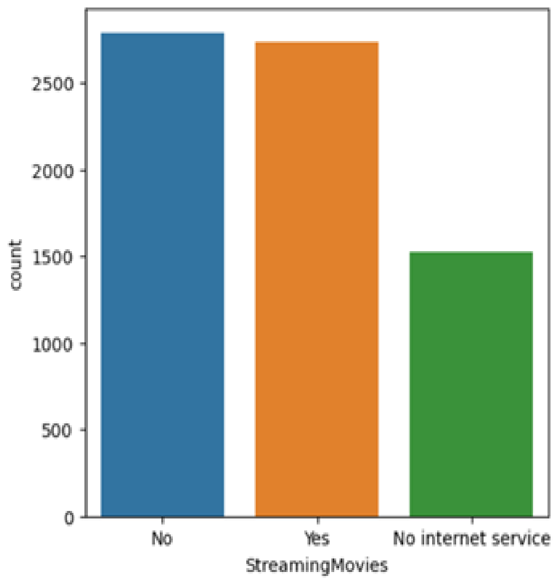
Streaming Tv

```
# StreamingTV :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.StreamingTV);
```



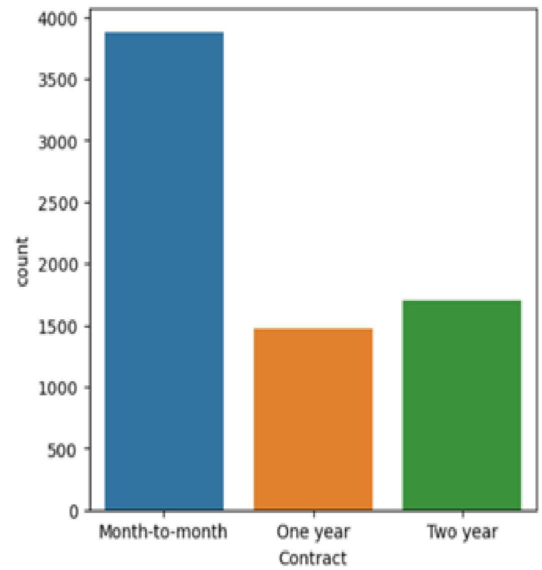
Streaming Movies

```
# StreamingMovies :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.StreamingMovies);
```



Contract

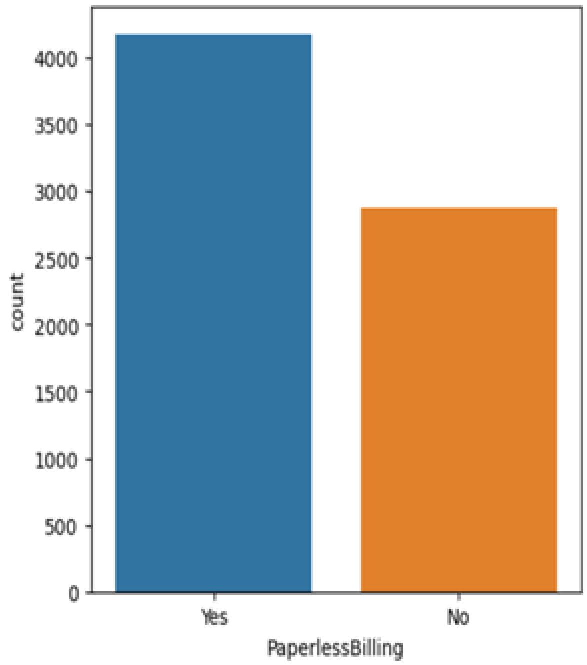
```
# Contract :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.Contract);
```



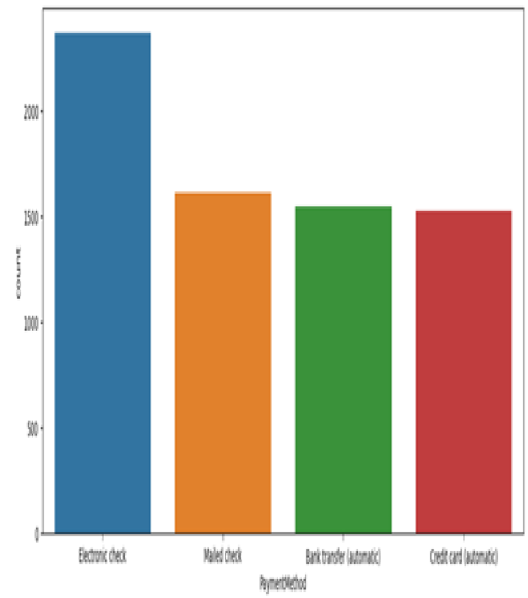
Paperless Billing

Payment method

```
# PaperlessBilling :

plt.figure(figsize=(5,5),dpi = 100)
sns.countplot(Customer_churn_data.PaperlessBilling);
```

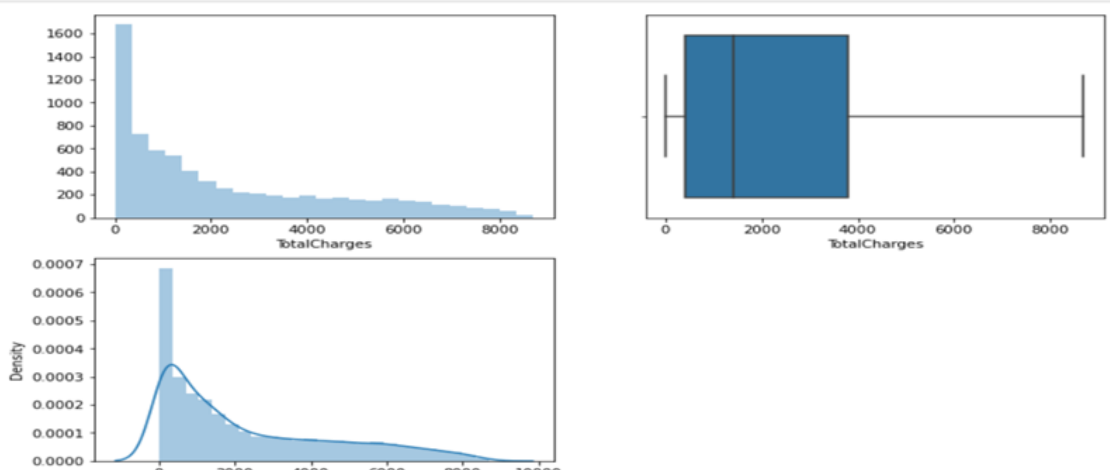



```
# PaymentMethod :

plt.figure(figsize=(15,5),dpi = 100)
sns.countplot(Customer_churn_data.PaymentMethod);
```



Total Charges

```
plt.figure(figsize=(12,8))
plt.subplot(2,2,1)
sns.distplot(Customer_churn_data['TotalCharges'], kde=False);
plt.subplot(2,2,2)
sns.boxplot(Customer_churn_data['TotalCharges']);
plt.subplot(2,2,3)
sns.distplot(Customer_churn_data['TotalCharges']);
```



OBSERVATIONS :

- Gender: Here we can see that both of the categories "Male" and "Female" are almost with same count.
- In the columns “Partner” and “Dependents” the category that has the highest count is “NO”, when compared to “Yes”.
- Tenure: Here we can see that the boxplot has no outliers present and the distribution curve is not at all normal .
- Phone Service: Here we can say that the column has the highest count for the category "yes" when compared to "No".
- In the columns “Multiple lines”, “Online security”, “Online Backup”, “Device protection”, “Tech support”, “Streaming TV”, “Streaming movies”.
- Internet Service: Here we can see that the column has highest count for the category "Fiber Optic “and the least count is for the category "No”.
- Contract: Here we can see that the column has the highest count for the category "Month-to-month" and the least count for the category "One year contract”.
- Paperless billing: Here we can see that the column has the highest count for the category "yes" than the other category "No”.
- Payment method: Here we can see that the column has the highest count for the category "Electronic check”, followed by the category "Mailed check" and both the categories Bank transfer and "Credit card" have more or less similar count.
- Total charges: Here we can see that the boxplot has no outliers and the distribution curve is high skewness towards right.

Before moving to correlation as our dataset contains the object values, we have to convert all those categorical values into numerical values through “Label Encoder”.

Here we have converted all the categorical variables into numerical and we can proceed with our heatmap plotting for correlation.

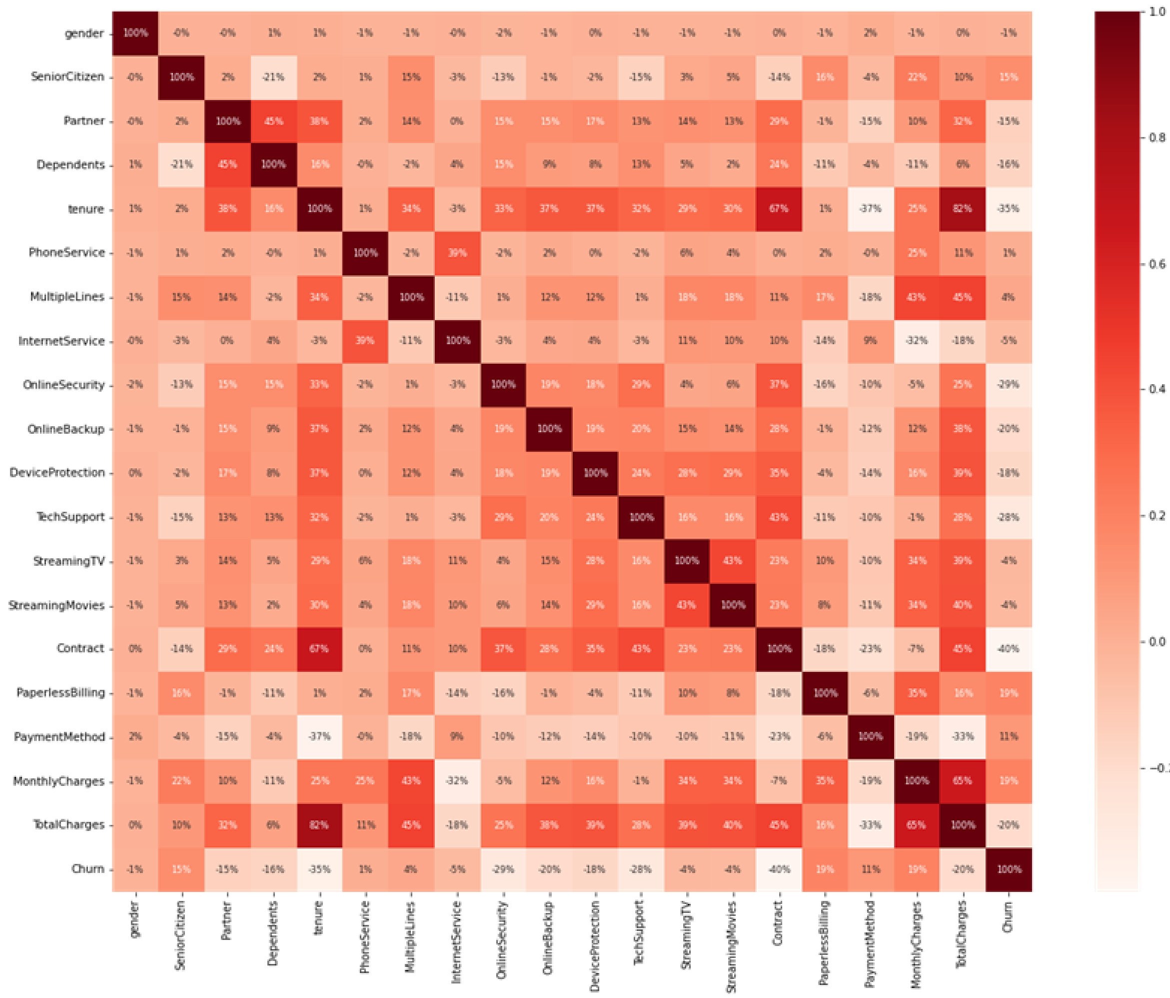
CORRELATION:

Here we will be going to check the relationship between the variables i.e.., how all these attributes are related with each other which means how strong is their bond with each other and also here we will be going to check how all these attributes correlate with our label column.

Also, with the help of this correlation we can also know the multicollinearity of the attributes if present which helps us to check through heat map plotting.

```
corr = Customer_churn_data.corr()
```

```
plt.figure(figsize = (20,15))
sns.heatmap(corr,annot = True,fmt = ".0%",cbar = True,square = True,annot_kws = {'size':8}, cmap = 'Reds')
plt.show()
```



OBSERVATIONS:

Here we can note that heatmap shows us the positive and negative relationships with each other and also with the label.

Here we have few features which are in positive relationship with each other and with the target variable but there are also few features which are in negative relation with each other and also with the target variable.

Here we can observe that the column "tenure" has the highest correlation with the column "total charges".

The columns like "Senior citizen", "Monthly charges", "paperless billing" and "payment method" have a positive correlation with our label column "Churn".

The columns like “contract”, “tenure”, “online security”, “techsupport”, “total charges”, “device protection”, “online backup”, “partner” and “dependents” have negative correlation with our label column “churn”.

The feature having very least positive correlation with the label is “phoneservice” and also the feature with least negative correlation with the label column is “gender”.

Also, we can note that no columns are multicollinear to each other i.e.., multicollinearity is not present in the data.

ESTIMATED DATA ANALYSIS CONCLUSION:

Through the analysis above we have analyzed that there were few columns which we have to convert their datatype for our better model building.

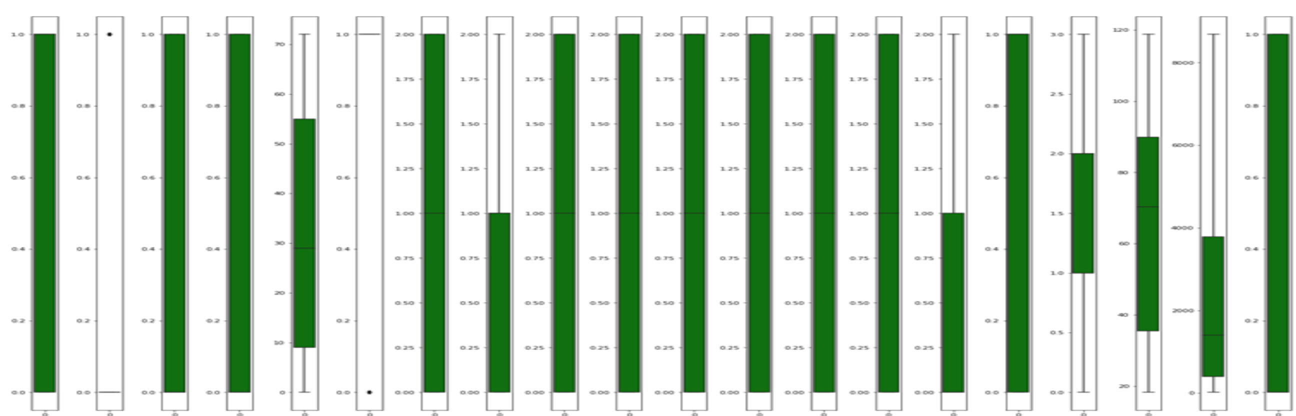
Majority of the customers are “Male” and most of the customers have opted for “Fiber optic internet service” but also there are customers with “no internet service”.

Many of the customers are with month-to-month contracts and also prefer paperless billing with electronic check as their payment method.

CHECKING FOR THE OUTLIERS:

Here we will know the columns with outliers.

In the above there are outliers seen in the column “Seniorcitizen” and probably there may be outliers further so it's better to treat them.



REMOVING THE OUTLIERS:

Here for removing the outliers we use the “Z-score method”.

```
from scipy.stats import zscore
z = np.abs(zscore(Customer_churn_data))
z.shape
```

```
(7043, 20)
```

```
threshold = 3
print(np.where(z>3))
```

```
(array([   0,    3,    7,   20,   27,   62,   81,   89,  103,  105,  107
         114,  116,  129,  131,  133,  168,  180,  185,  187,  206,  211,
         215,  216,  217,  225,  236,  252,  255,  259,  260,  263,  272,
         278,  303,  321,  324,  328,  348,  354,  358,  372,  376,  382,
         387,  398,  424,  431,  435,  452,  465,  481,  488,  495,  498,
         544,  569,  596,  610,  616,  620,  634,  660,  667,  669,  674,
         677,  688,  716,  718,  735,  765,  776,  784,  790,  794,  813,
         829,  843,  847,  859,  866,  873,  875,  877,  884,  893,  917,
         934,  941,  943,  960,  973, 1011, 1018, 1037, 1050, 1051, 1053,
        1072, 1110, 1119, 1122, 1144, 1146, 1150, 1161, 1169, 1182, 1204,
        1221, 1225, 1242, 1255, 1257, 1271, 1278, 1298, 1311, 1326, 1331,
        1333, 1334, 1340, 1349, 1352, 1365, 1379, 1402, 1407, 1416, 1452,
        1479, 1480, 1481, 1500, 1506, 1513, 1519, 1560, 1562, 1581, 1584,
        1614, 1620, 1634, 1637, 1652, 1689, 1692, 1694, 1703, 1722, 1734,
        1789, 1802, 1803, 1819, 1827, 1832, 1845, 1851, 1854, 1862, 1881,
        1889, 1892, 1894, 1906, 1910, 1944, 1959, 1969, 1985, 1989, 1998,
        2002, 2031, 2046, 2050, 2087, 2089, 2090, 2117, 2124, 2127, 2131,
        2188, 2215, 2225, 2226, 2237, 2239, 2290, 2295, 2310, 2340, 2344,
        2348, 2362, 2382, 2383, 2385, 2398, 2399, 2409, 2412, 2413, 2417,
        2420, 2421, 2426, 2427, 2431, 2433, 2465, 2468, 2492, 2533, 2538,
        2541, 2547, 2562, 2608, 2610, 2626, 2637, 2644, 2661, 2662, 2681,
        2696, 2700, 2709, 2712, 2718, 2725, 2728, 2748, 2751, 2752, 2754,
        2761, 2773, 2781, 2804, 2809, 2814, 2841, 2842, 2889, 2898, 2899,
        2903, 2913, 2915, 2916, 2918, 2919, 2929, 2940, 2944, 2962, 2966,
        2972, 2990, 2992, 2994, 2995, 3020, 3028, 3036, 3039, 3042, 3043,
        3060, 3062, 3070, 3073, 3080, 3092, 3096, 3126, 3127, 3133, 3139,
        3150, 3160, 3174, 3177, 3183, 3185, 3190, 3191, 3194, 3213, 3221,
        3223, 3233, 3235, 3243, 3258, 3290, 3292, 3311, 3316, 3318, 3342,
```

```
Customer_churn_data_new = Customer_churn_data[(z<3).all(axis = 1)]
print(Customer_churn_data.shape)
print(Customer_churn_data_new.shape)
```

```
(7043, 20)
(6361, 20)
```

Here through Z-Score method we have tried to treat the outliers present in the dataset and the threshold number here used is 3 and also, we have assigned a new variable to the data after using threshold along with which we have printed the shapes of both the datasets, the previous dataset and the new one, out of which we can see that the new dataset is with less number of records than the previous dataset and the loss percentage which is calculated is 9.68%.

CHECKING THE SKEWNESS:

Here we can see that our column “Total charges” reduced its skewness.

```
Customer_churn_data_new.skew()
```

```
gender             -0.014781
SeniorCitizen       1.823376
Partner             0.056316
Dependents          0.876594
tenure              0.237945
PhoneService        0.000000
MultipleLines       0.132058
InternetService     0.049126
OnlineSecurity      0.422032
OnlineBackup        0.167910
DeviceProtection    0.183254
TechSupport         0.409833
StreamingTV        -0.002734
StreamingMovies    -0.010025
Contract            0.629701
PaperlessBilling   -0.386613
PaymentMethod      -0.169889
MonthlyCharges     -0.399139
TotalCharges        0.899649
Churn               1.053055
dtype: float64
```

```python
features = ["TotalCharges","SeniorCitizen"]
```

```python
from sklearn.preprocessing import PowerTransformer
scaler = PowerTransformer(method='yeo-johnson')
'''
parameters:
method = 'box-cox' or 'yeo-johnson'
'''
```

```
"\nparameters:\nmethod = 'box-cox' or 'yeo-johnson'\n"
```

```python
Customer_churn_data_new[features] = scaler.fit_transform(Customer_churn_data_new[features].values)
Customer_churn_data_new[features]
```

	TotalCharges	SeniorCitizen
1	0.222457	-0.441591
2	-1.400891	-0.441591
4	-1.265537	-0.441591
5	-0.379577	-0.441591
6	0.247579	-0.441591
...	...	...
7037	0.001314	-0.441591
7038	0.264486	-0.441591
7039	1.527177	-0.441591
7041	-0.942361	2.264538
7042	1.445003	-0.441591

6361 rows × 2 columns

```
Customer_churn_data_new.skew()
```

```
gender              -0.014781
SeniorCitizen        1.823376
Partner              0.056316
Dependents           0.876594
tenure               0.237945
PhoneService         0.000000
MultipleLines        0.132058
InternetService      0.049126
OnlineSecurity       0.422032
OnlineBackup         0.167910
DeviceProtection     0.183254
TechSupport          0.409833
StreamingTV         -0.002734
StreamingMovies     -0.010025
Contract             0.629701
PaperlessBilling    -0.386613
PaymentMethod       -0.169889
MonthlyCharges      -0.399139
TotalCharges        -0.148971
Churn                1.053055
dtype: float64
```

DATA PRE-PROCESSING:

Separating the independent and traget variables :

train_test_split :

```
x = Customer_churn_data_new.drop("Churn", axis=1)
y = Customer_churn_data_new["Churn"]
```

Here we have divided the dataset into two variables: x and y, in which "x" is variable to which all the features are assigned except the label column "Churn" which is assigned to the variable "y".

```
x.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSuppc
1	1	-0.441591	0	0	34	1	0	0	2	0	2	
2	1	-0.441591	0	0	2	1	0	0	2	2	0	
4	0	-0.441591	0	0	2	1	0	1	0	0	0	
5	0	-0.441591	0	0	8	1	2	1	0	0	2	
6	1	-0.441591	0	1	22	1	2	1	0	2	0	

```
y.head()
```

```
1    0
2    1
4    1
5    1
6    0
Name: Churn, dtype: int32
```

Scaling the x_data using the "Standard Scalar":

Here we have scaled the x_data ie., feature data through "StandardScaler" and then we have imported train_test_split and divided the dataset into x_train, y_train, x_test, y_test with test size = 0.3 which means 70% of the train data is taken and 30% of the test data is considered with random state (it's a random guess of any number).

Building the Machine Learning Models:

```
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import  RandomForestClassifier , GradientBoostingClassifier
```

Training :

```
models = {"LogisticRegression" : LogisticRegression(),
          "K-Nearest Neighbors": KNeighborsClassifier(),
          "Decision Tree"      : DecisionTreeClassifier(),
          "Random Forest"      : RandomForestClassifier(),
          "Gradient Boosting"  : GradientBoostingClassifier()}

for name, model in models.items():
    model.fit(x_train,y_train)
    print(name + " is trained now.")
```

```
LogisticRegression is trained now.
K-Nearest Neighbors is trained now.
Decision Tree is trained now.
Random Forest is trained now.
Gradient Boosting is trained now.
```

Here we have imported the necessary libraries required for model building and I have used forloop for training and testing the models.

Testing :

```
for name, model in models.items():
    print(name + ": {:,.2f}%".format(model.score(x_test,y_test)*100))
```

```
LogisticRegression: 80.46%
K-Nearest Neighbors: 76.58%
Decision Tree: 73.44%
Random Forest: 79.94%
Gradient Boosting: 80.78%
```

In testing all the models, the "Gradient Boosting" model has the highest test score compared to all the other models.

HYPER PARAMETER TUNING:

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = {'criterion':['mse', 'mae'],
              'max_features':['auto', 'sqrt', 'log2'],
              'n_estimators':[0,40],
              'max_depth':[2,4,6]}
```

Gradient Boosting Classifier :

here we use "GradientBoostingClassifier" because this model has highest accuracy score when compared to the other models.

```
GCV=GridSearchCV(GradientBoostingClassifier(),parameters,cv=5)
```

```
GCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=GradientBoostingClassifier(),
             param_grid={'criterion': ['mse', 'mae'], 'max_depth': [2, 4, 6],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'n_estimators': [0, 40]})
```

```
GCV.best_params_
```

```
{'criterion': 'mse',
 'max_depth': 4,
 'max_features': 'auto',
 'n_estimators': 40}
```

Here I have selected the model "Gradient Boosting model" which has the highest accuracy percent among all the other models for hyper parameter tuning.

For this I have selected a few parameters for tuning our model which are: Criterion, max_depth, max_features, n_estimators and got the best parameters for tuning.

Here I can see that the cross-validation score is 80.42%.
After tuning the model accuracy decreased to certain extent.

CLASSIFICATION REPORT:

```
from sklearn.metrics import accuracy_score, classification_report, roc_auc_score
```

```
print(classification_report(y_test,pred_decision))
```

```
              precision    recall  f1-score   support

           0       0.85      0.90      0.88      1430
           1       0.64      0.54      0.59       479

    accuracy                           0.81      1909
   macro avg       0.75      0.72      0.73      1909
weighted avg       0.80      0.81      0.80      1909
```

Observation : Here we can see that the model is good and f1 score is balanced .

Checking for the AUC Score :

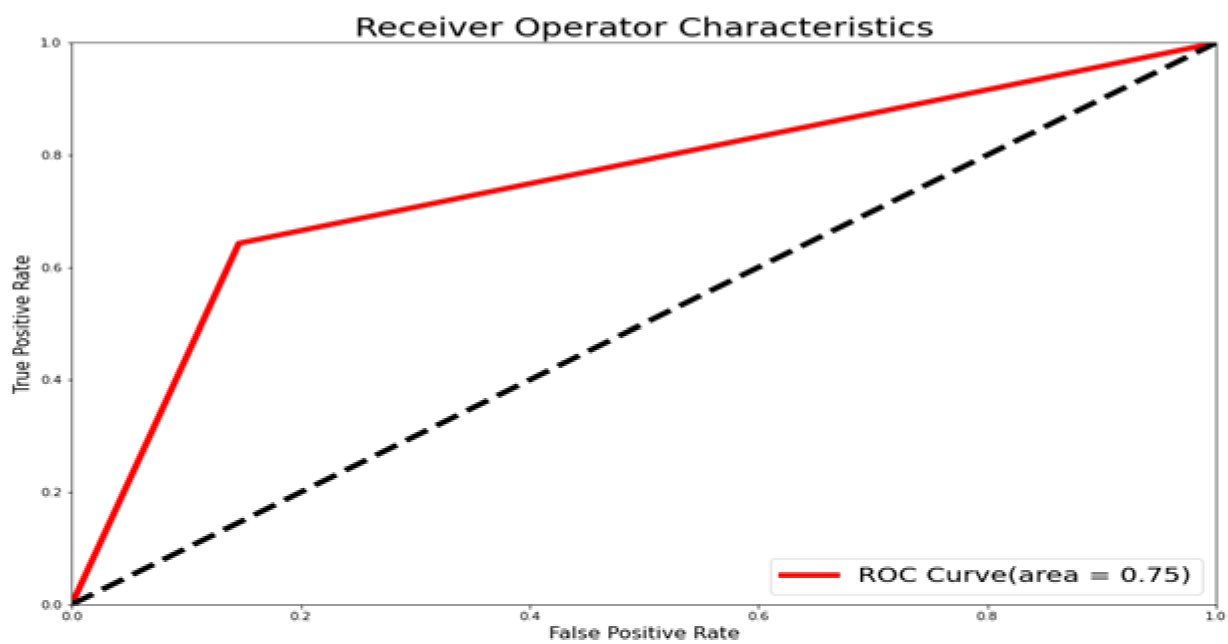
```
print(roc_auc_score(y_test, pred_decision))
```

```
0.7200052557046295
```

Here I have checked the auc_score which is 72%.

PLOTTING THE ROC CURVE:

```
from sklearn.metrics import roc_curve, auc
fpr,tpr, thresholds = roc_curve(pred_decision, y_test)
roc_auc = auc(fpr,tpr)
plt.figure(figsize = (15,10))
plt.plot(fpr, tpr, lw=5, color = 'red',label = 'ROC Curve(area = %0.2f)'%roc_auc)
plt.plot([0,1],[0,1], lw =5, color = 'black',linestyle = '--')
plt.xlim(0.0,1.0)
plt.ylim(0.0,1.0)
plt.xlabel('False Positive Rate', fontsize = 15)
plt.ylabel('True Positive Rate', fontsize = 15)
plt.title('Receiver Operator Characteristics',fontsize = 25)
plt.legend(loc = 'lower right', fontsize = 20)
plt.show()
```



Here I have plotted the ROC Curve for our best model is 75%

SAVING THE MODEL:

```
import pickle
filename = 'churn.pkl'
pickle.dump(GradientBoosting,open(filename, 'wb'))
```

```
loaded_model = pickle.load(open("churn.pkl", "rb"))
result = loaded_model.score(x_test, y_test)
print(result)
```

```
0.8093242535358827
```

Here I have saved the model and also checked the test score which is 80.9% ie., almost 81%.

PREDICTING THE CHURNED VALUES OF THE CUSTOMERS:

```
conclusion = pd.DataFrame([loaded_model.predict(x_test)[:],pred_decision[:]],index = ["Predicted","Original"])
```

```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	1899	1900	1901	1902	1903	1904	1905	1906	1907	1908
Predicted	0	0	1	0	1	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
Original	0	0	1	0	1	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0

2 rows × 1909 columns

Hence we can conclude saying that our best model "Gradient Boosting" is with accuracy achieved is "80.7%"

CONCLUSION:

This problem needs a good vision on data, and in this problem "Feature Engineering" is the most crucial thing. You can see how we handled numerical and categorical data and also how we build different machine learning models on the same dataset.